FIWARE
Open Apis for Open Minds

# Usando FIWARE

# FIWARE ZONE_

http://www.fiware.org
http://lab.fiware.org
Follow @FIWARE on Twitter!

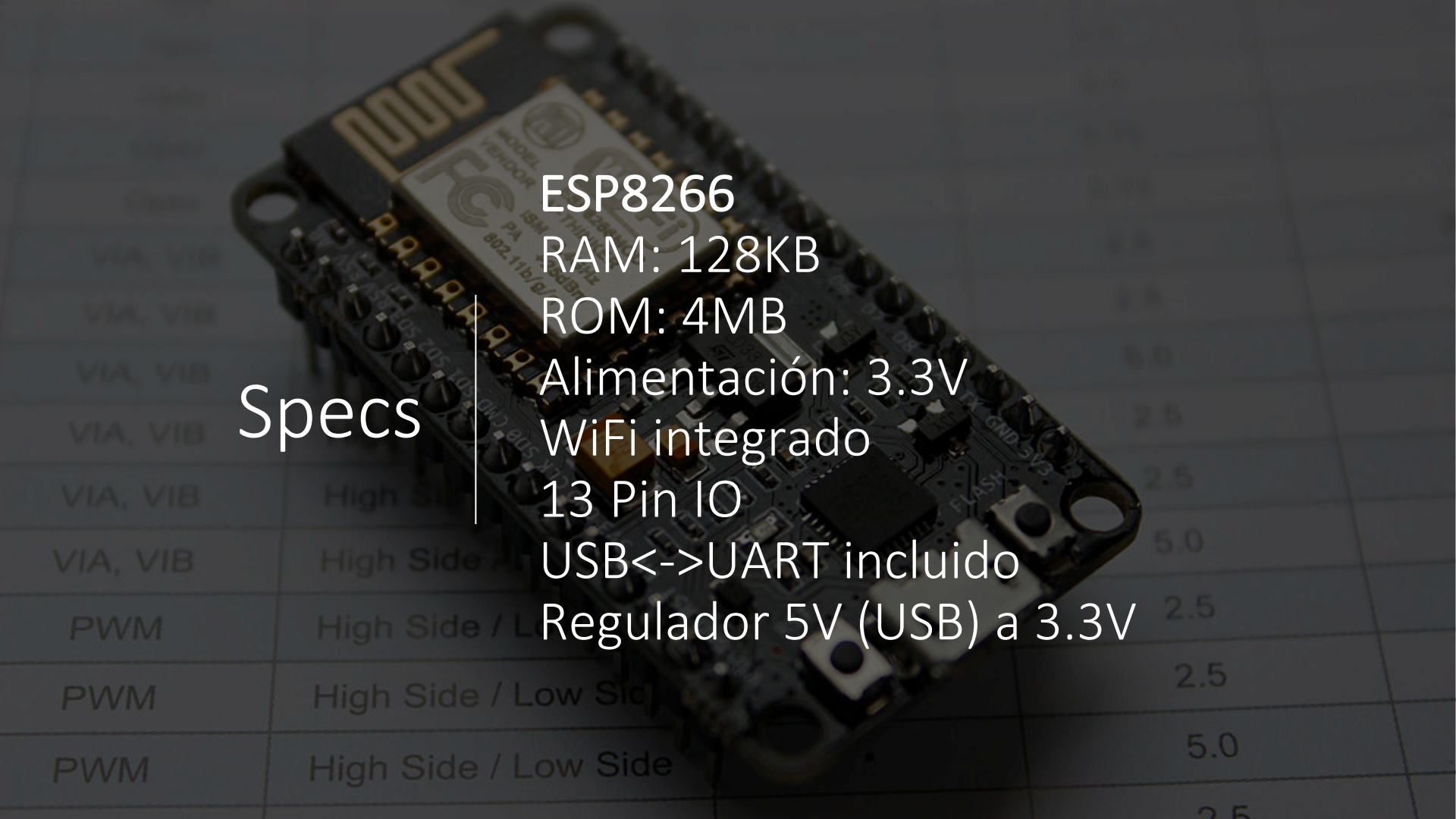
FIWARE ZONE

# 0. Conocimientos previos

1. C++
2. HTML
3. API REST
4. FIWARE – IoT Agent Ultraligth 2.0

# 01

# Hardware



# Specs

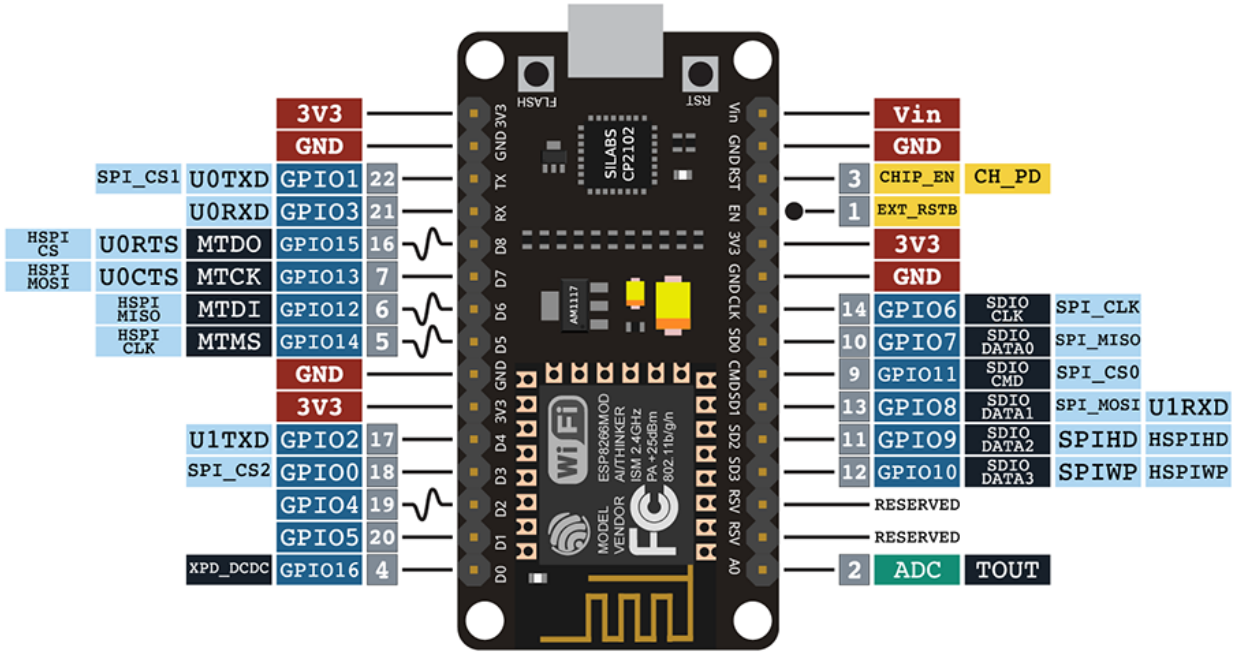
ESP8266
RAM: 128KB
ROM: 4MB
Alimentación: 3.3V
WiFi integrado
13 Pin IO
USB<->UART incluido
Regulador 5V (USB) a 3.3V

# ESP-12E DEVELOPMENT BOARD
## PINOUT

NOTES:
- Typ. pin current 6mA (Max. 12mA)
- For sleep mode, connect GPIO16 and EXT_RSTB. On wakeup, GPIO16 will output LOW for system reset.
- On boot/reset/wakeup, keep GPIO15 LOW and GPIO2 HIGH.

ACROBOTIC
https://acrobotic.com
CC BY SA
03-31-2016
v1.0

FIWARE ZONE




POWER
I/O
ADC
CONTROL
N/C
SP. FUNCTION(S)
COMM. INTERFACE
PIN NUMBER
PWM

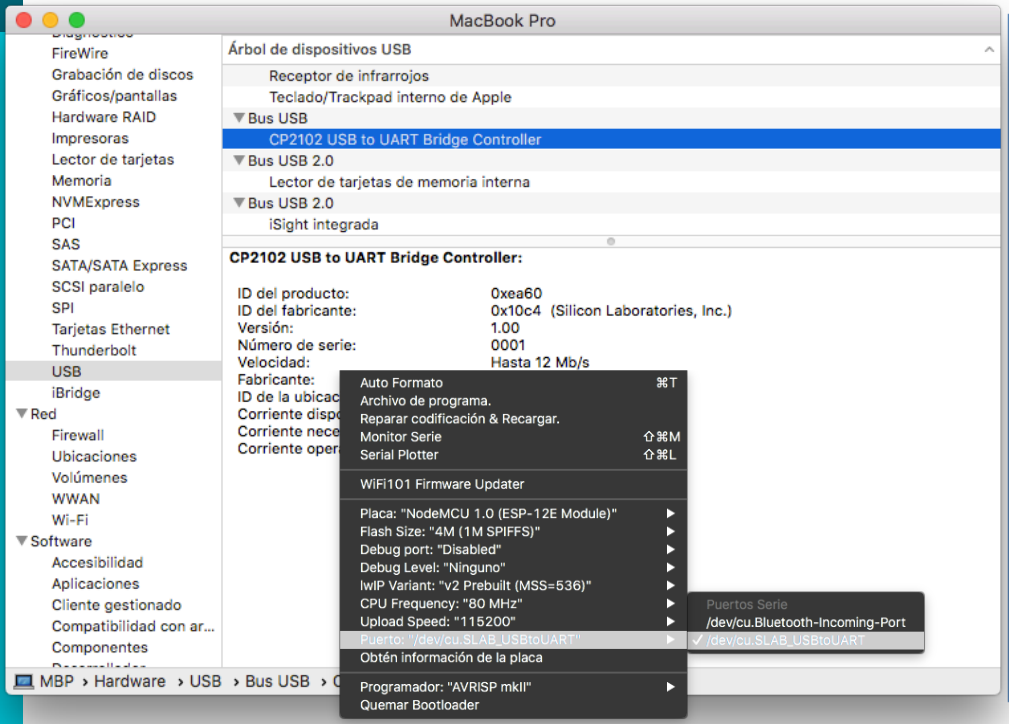
Telefónica

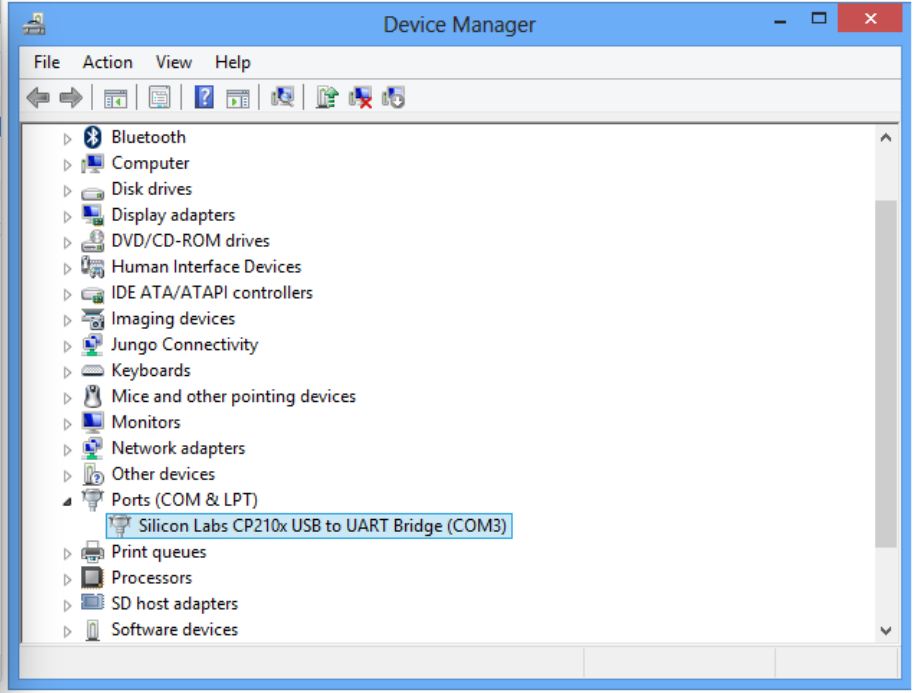
JUNTA DE ANDALUCIA

# 02

# Instalación e IDE

# 2.1 Drivers

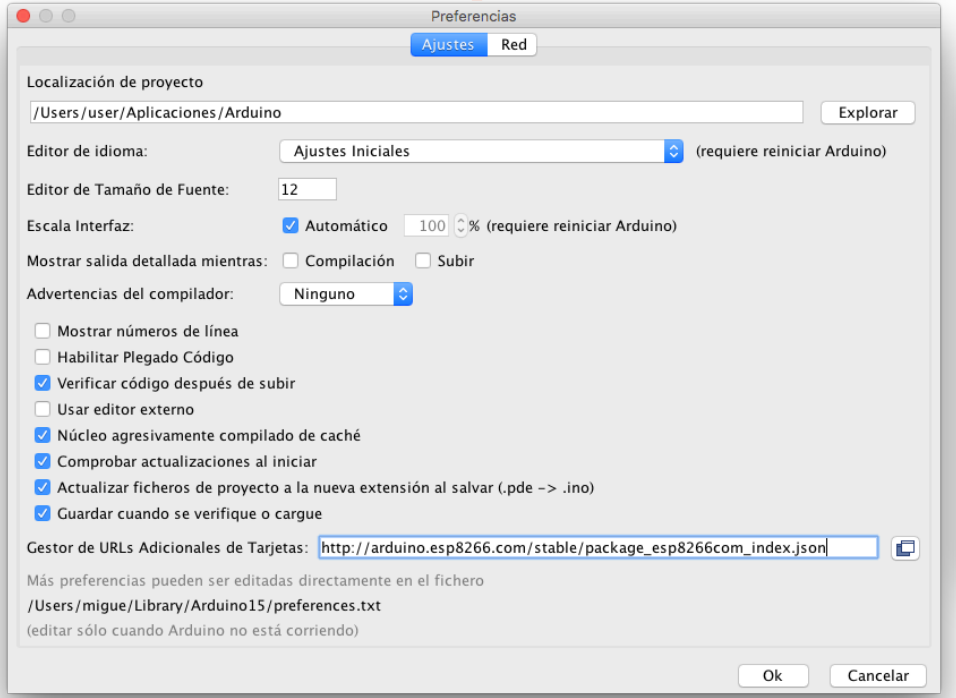


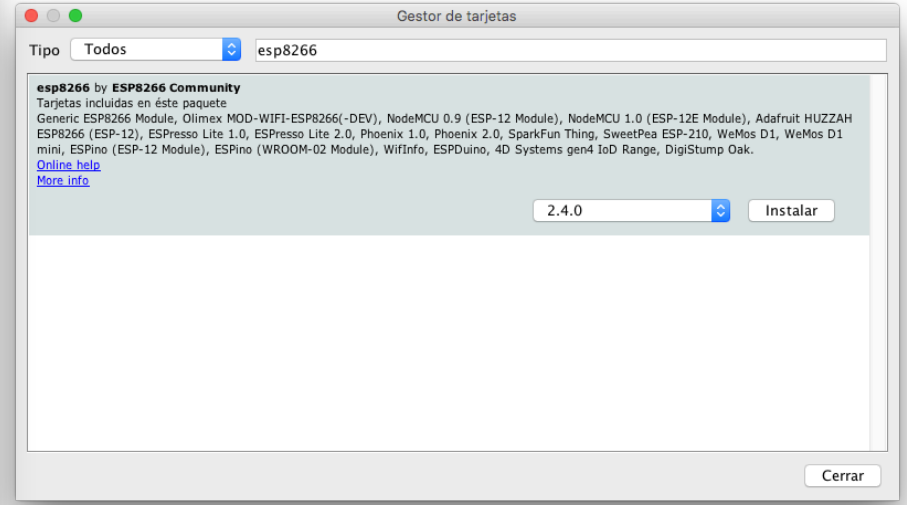

www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers

# 2.2 Arduino

1

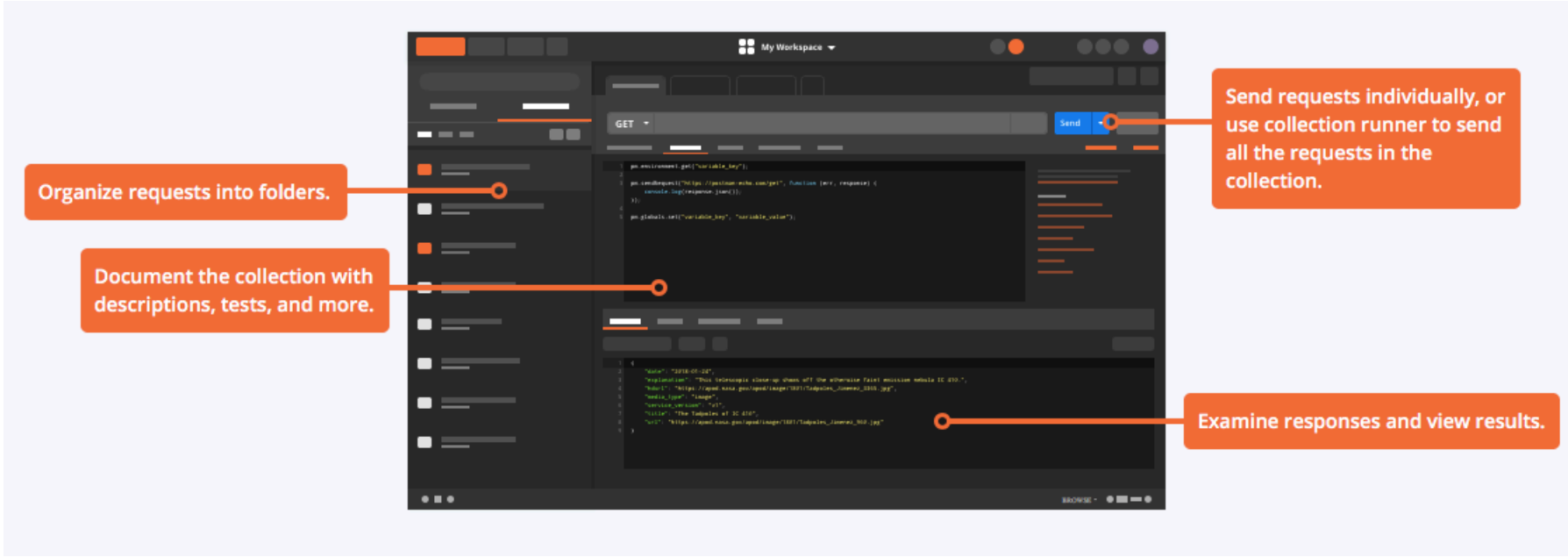



2



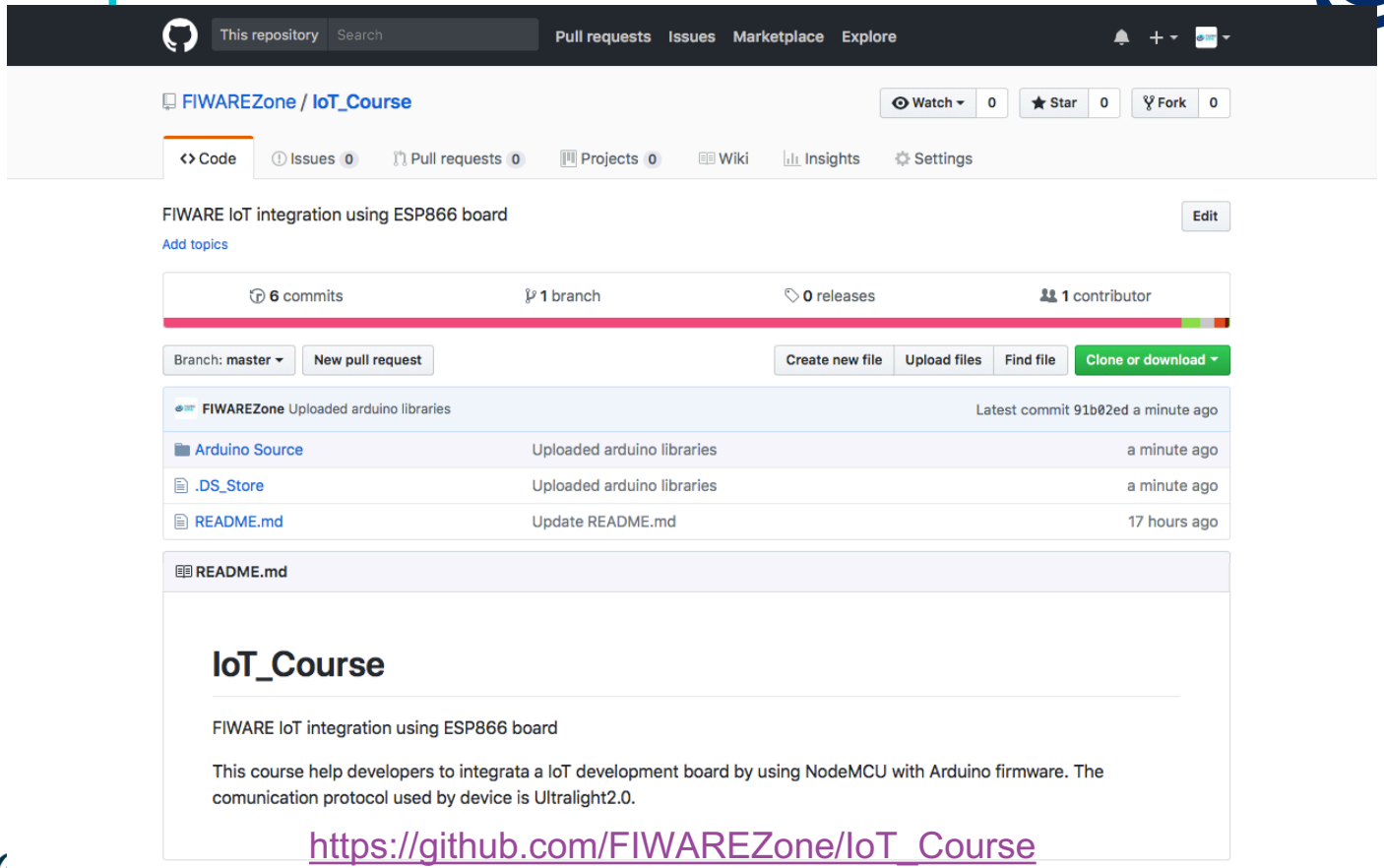

http://arduino.esp8266.com/stable/package_esp8266com_index.json

# 2.3 Postman




https://www.getpostman.com

# 2.4 Repositorio del curso




https://github.com/FIWAREZone/IoT_Course

# 03

# Configurando dispositivo en FIWARE

# 3.5 Creando la entidad

**HTTP POST:**

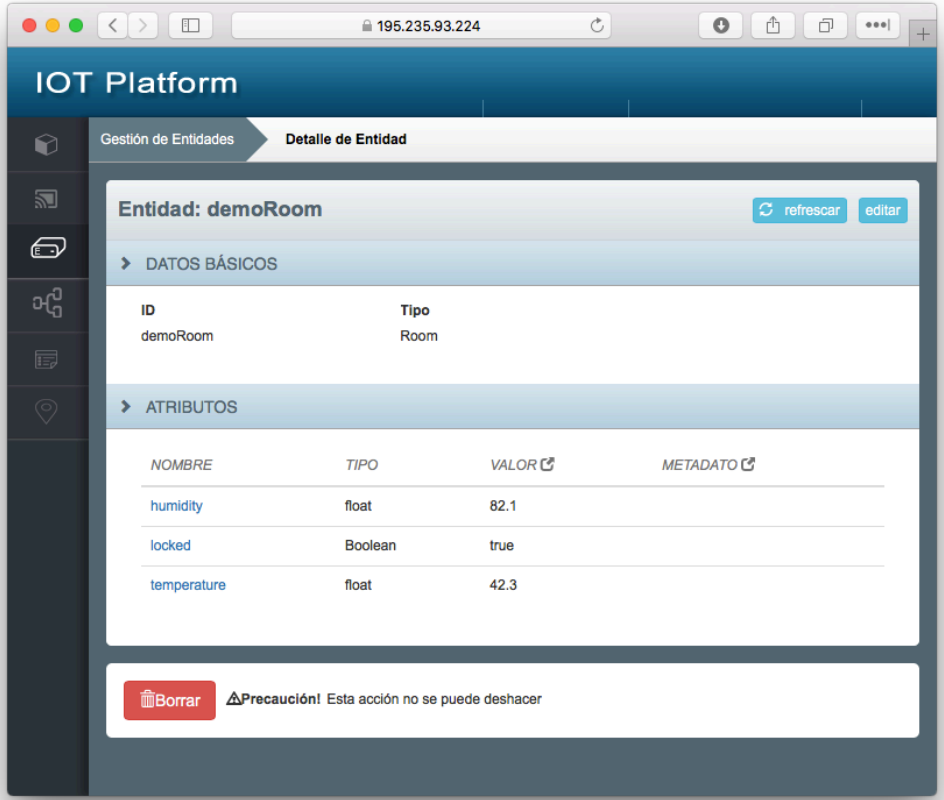
https://{{host}}:10027/v2/entities

**Headers:** {'content-type': 'application/json'; "Fiware-Service: **{{service}}**"; "Fiware-ServicePath:**{{subservice}}**"; "X-Auth-Token: **{{token}}**"}

**Payload:**

```
{
        "id": "demoRoom",
        "type": "Room",
        "locked": {
                "value": true,
                "type": "Boolean"
        },
        "temperature": {
                "value": 42.3,
                "type": "float"
        },
        "humidity": {
                "value": 82.1,
                "type": "float"
        }

}
```

# 3.5 Creando la entidad



# 3.5 Creando el dispositivo

**HTTP POST:**

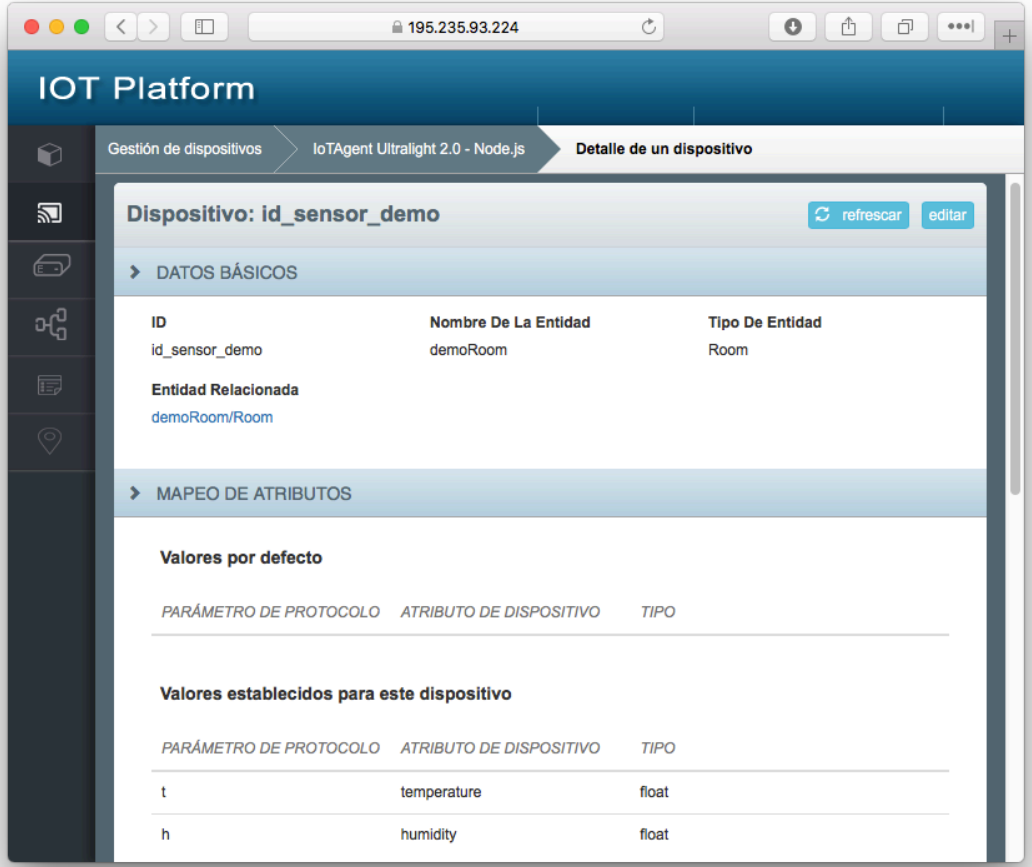
https://{{host_iota}}:8088/iot/devices

**Headers:** {'content-type': 'application/json'; "Fiware-Service: **{{service}}**"; "Fiware-ServicePath:**{{subservice}}**"; "X-Auth-Token: **{{token}}**"}

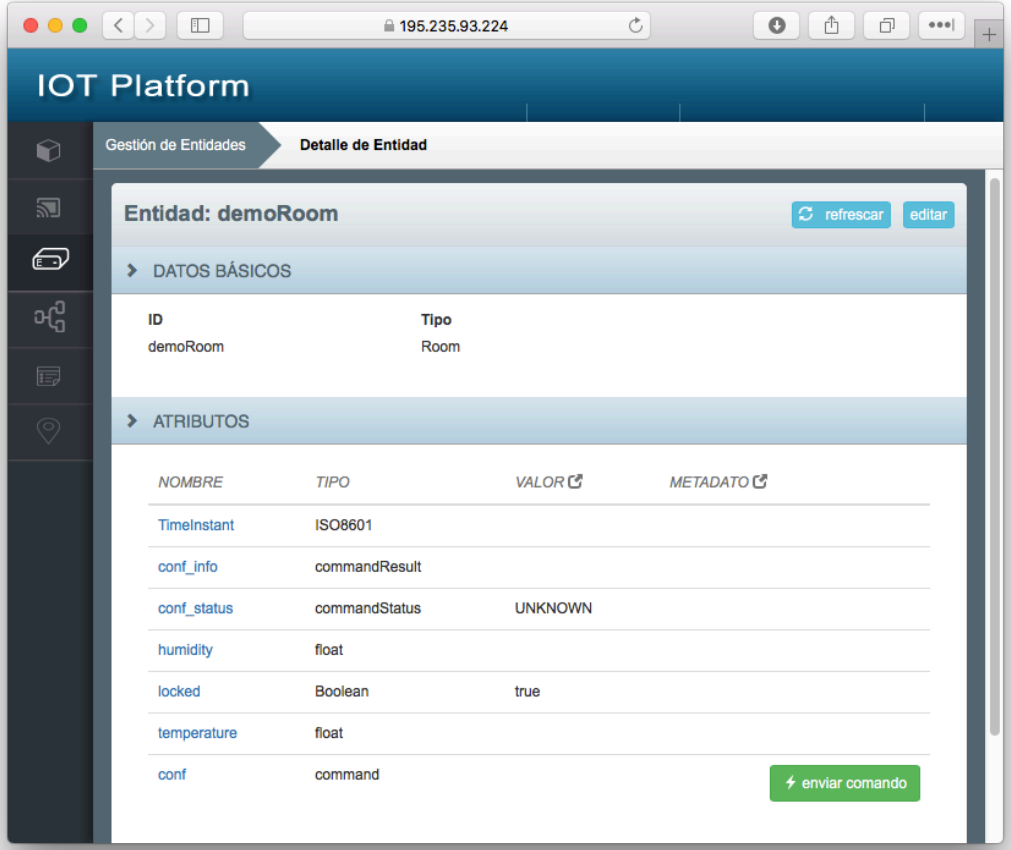
**Payload:**

```
{
   "devices": [
      {
         "device_id": "id_sensor_demo",
         "entity_name": "demoRoom",
         "entity_type": "Room",
         "attributes": [
             { "object_id": "t", "name": "temperature", "type": "float" },
             { "object_id": "h", "name": "humidity", "type": "float" }
         ],
         "lazy":[ ],
         "commands": [
             { "object_id": "conf", "name": "conf", "type": "command" }
         ],
         "static_attributes": [ ],
     "protocol": "IoTA-UL",
     "transport": "HTTP"
      }
   ]
}
```

# 3.5 Creando el dispositivo



# 3.5 Creando el dispositivo



# 3.5 Enviando datos por UL

**HTTP POST:**

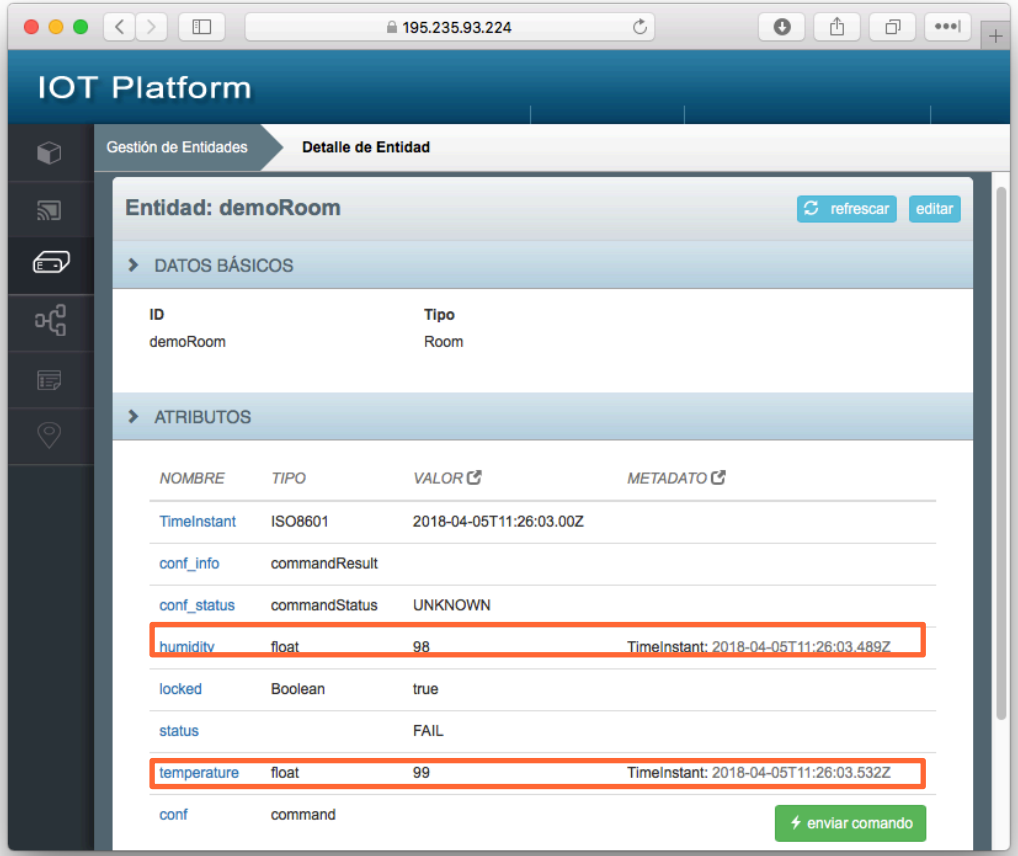
http://{{host_iota}}:8085/iot/d?k={{UL_apikey}}&i={{device_ID}}

**Headers:** {'content-type': text/plain'}

**Payload:**

t|99#h|98

# 3.5 Enviando datos por UL



# 3.5 Enviando datos al dispositivo

**HTTP POST:**

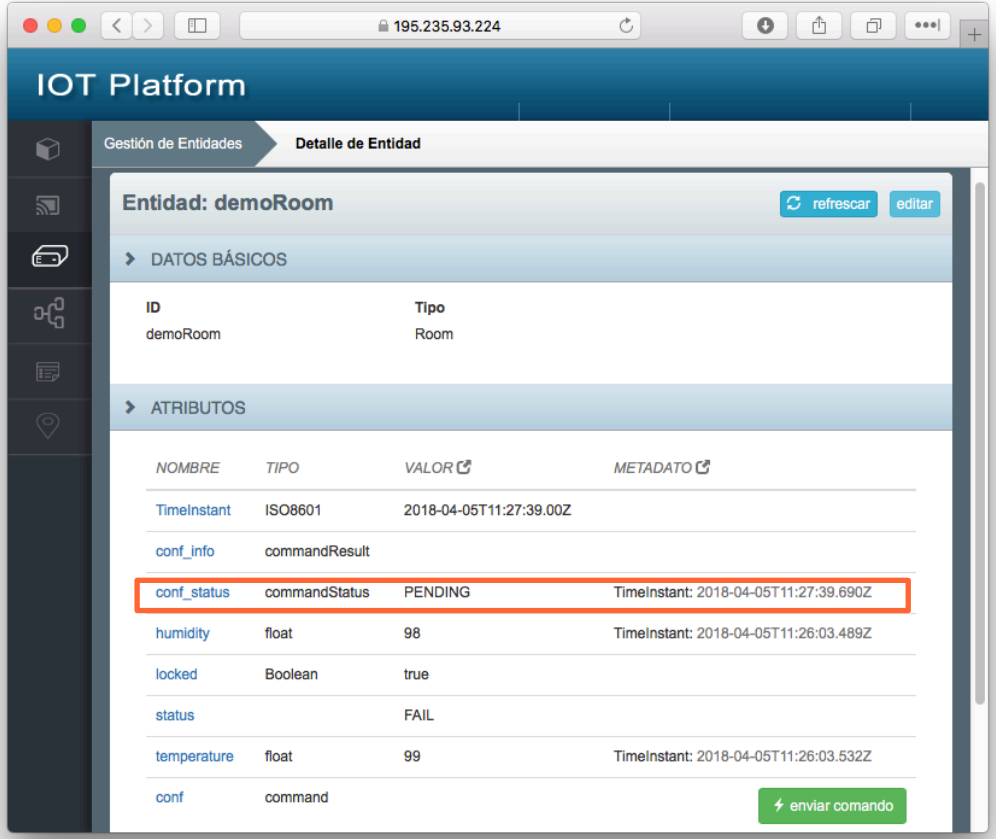
https://{{host}}:10027/v1/contextEntities/type/{{device_type}}/id/{{entity_name}}/attributes/conf

**Headers:** {'content-type': 'application/json'; "Fiware-Service: **{{service}}**"; "Fiware-ServicePath:**{{subservice}}**"; "X-Auth-Token: **{{token}}**"}

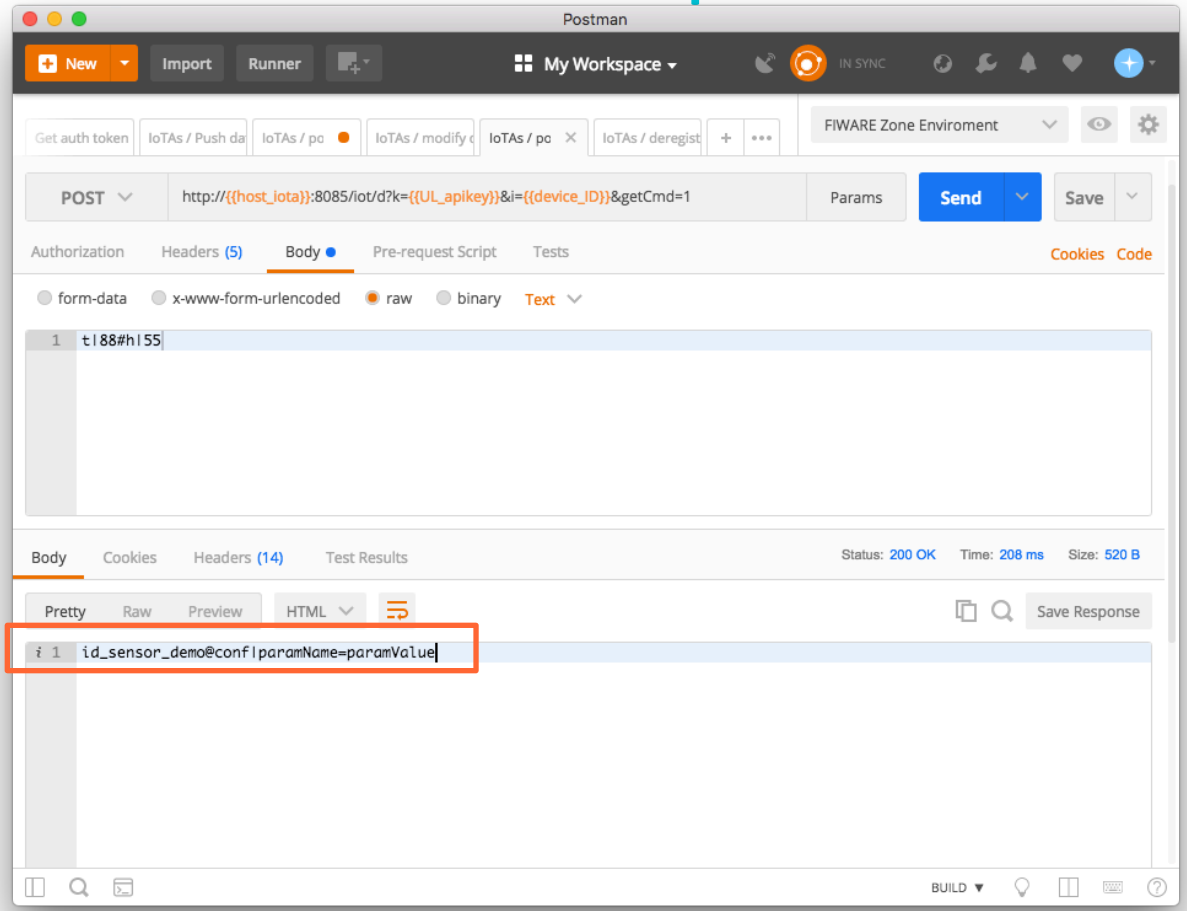
**Payload:**

{"value":{"paramName":"paramValue"},"type":"command"}

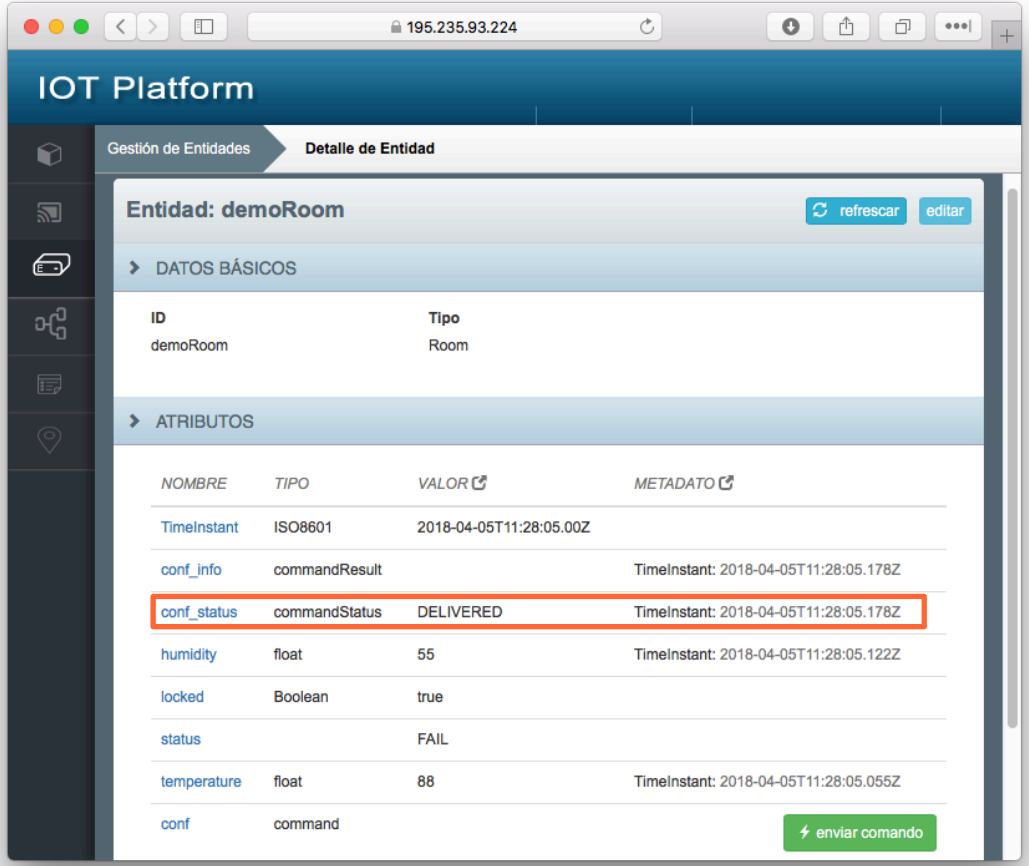
# 3.5 Enviando datos al dispositivo



# 3.5 Enviando datos al dispositivo



# 3.5 Enviando datos al dispositivo



# 04

# Programa y funcionamiento

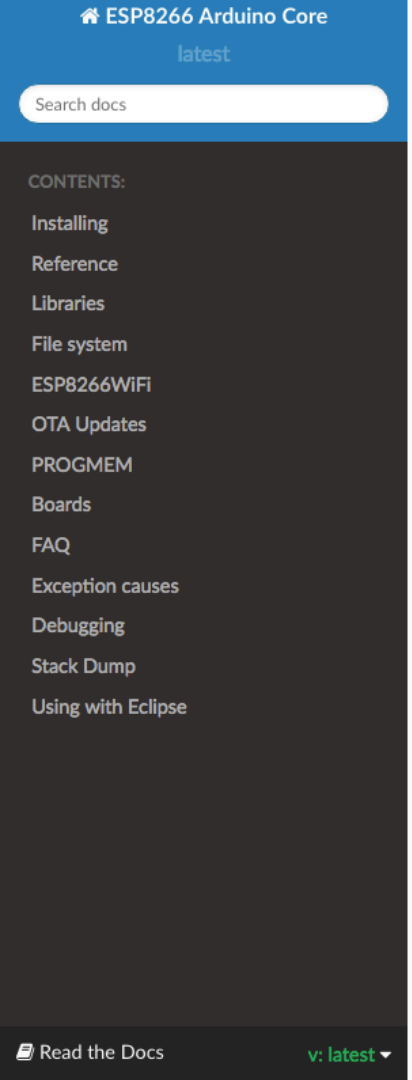
# 4.2 Variables de configuración

```
/*************** Global Variables ***************/
//FIWARE Variables
String FIWARE_ID = "REEMPLAZAR_ID";
String FIWARE_server = "195.235.93.235";
String FIWARE_port = "8085";
String FIWARE_token = "REEMPLAZAR_APIKEY";

//WiFi Network Names
const char* WiFi_SoftAP_Name = "FIWAREZone_IoT";
const char* WiFi_SoftAP_WiFi_Name = "FIWAREZone_IoT_Wifi";

//OTA
const char* update_path = "/webota";
const char* update_username = "admin";
const char* update_password = "admin";
```

# 4.2 Librerías empleadas

```
//ESP8266 Native libraries
#include <ESP8266WiFi.h>          //https://github.com/esp8266/Arduino
#include <DNSServer.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
#include <ESP8266HTTPUpdateServer.h>
#include <ESP8266HTTPClient.h>

//External libraries
#include <WiFiManager.h>          //https://github.com/tzapu/WiFiManager
#include <ArduinoJson.h>          //https://github.com/bblanchon/ArduinoJson
```

# 4.2.1 Librerías nativas ESP8266 (WiFi)



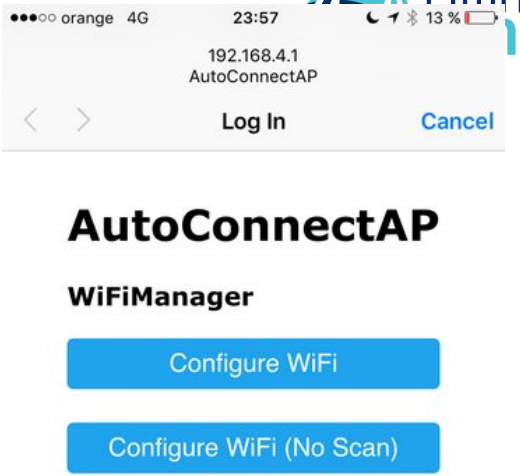

- ESP8266AVRISP
- ESP8266HTTPClient
- ESP8266HTTPUpdateServer
- ESP8266LLMNR
- ESP8266NetBIOS
- ESP8266SSDP
- ESP8266WebServer
- ESP8266WiFi
- ESP8266WiFiMesh
- ESP8266httpUpdate
- ESP8266mDNS

https://github.com/esp8266/Arduino

# 4.2.2 Librerías externas

**WiFiManager**

Permite configurar un punto de acceso
Si no lo encuentra, crea red propia para configurar conectándote mediante wifi
Guarda las credenciales en la memoria interna




https://github.com/tzapu/WiFiManager

# 4.2.2 Librerías externas

**ArduinoJson**

Serialización y deserialización de JSON de forma sencilla para el envío de datos

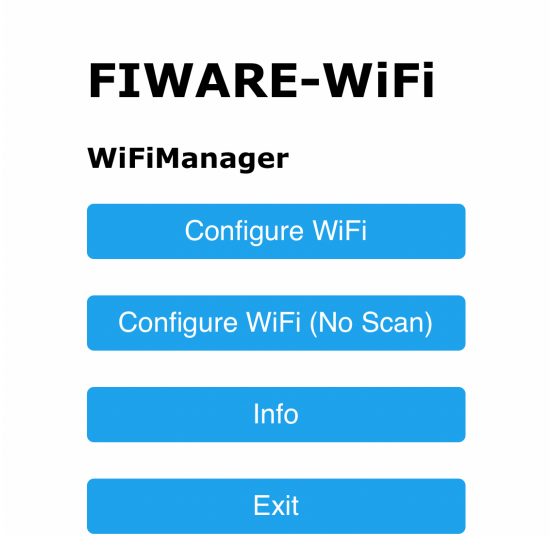

https://github.com/bblanchon/ArduinoJson

# 4.3 Configuración WiFi


192.168.4.1

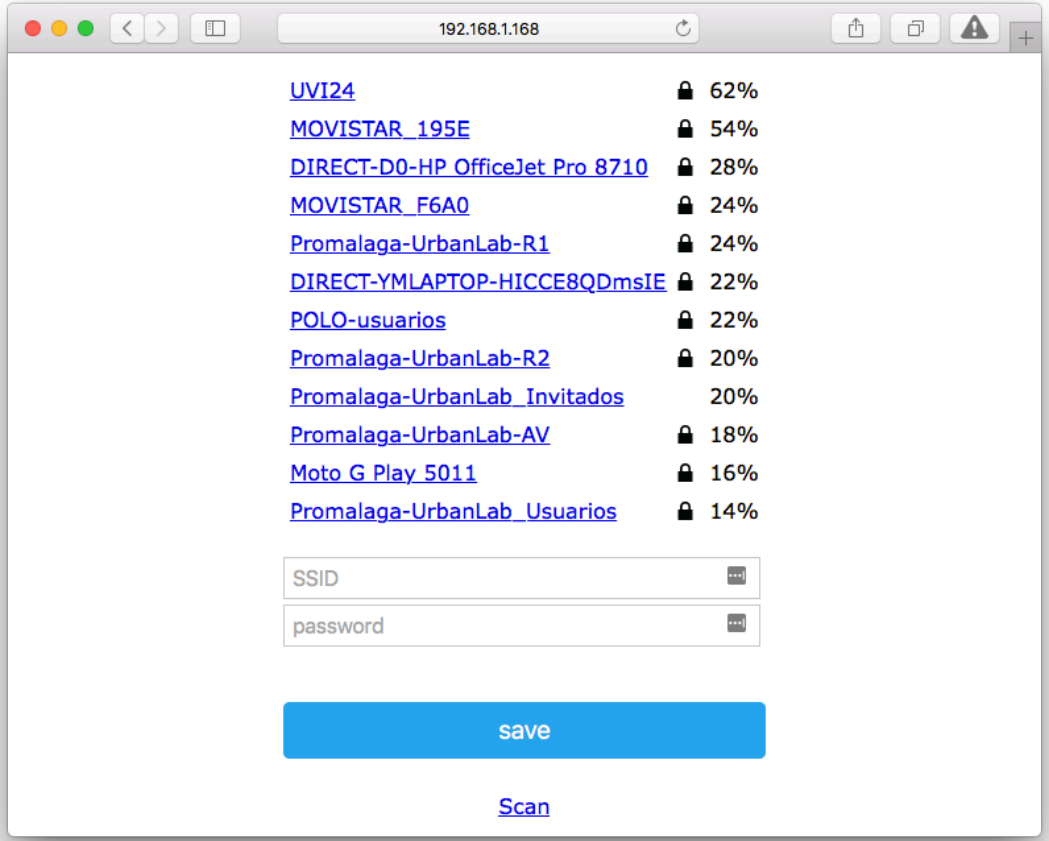
# 4.3 Configuración WiFi


Portal cautivo, entrar en “Configure WiFi” e introducir los datos de la red wifi



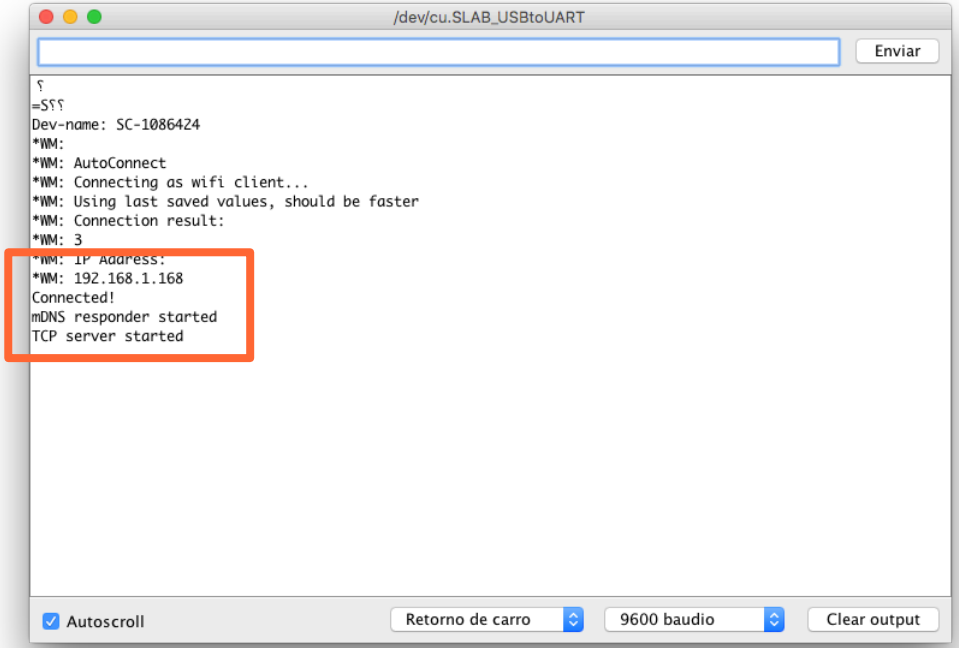

192.168.4.1

# 4.3 Configuración WiFi



# 4.3 Configuración WiFi

Revisamos el monitor del puerto serie
Ya está conectado a la Wifi que hemos configurado. Ahora podemos acceder desde el navegado, conectado a la misma red



# 4.4 Código fuente

**FIWARE.ino**
void setup()

```
//Bind Webserver URL functions
server.on ( "/", handleRoot );
server.on ( "/wrst", handleWrst );
server.on ( "/wifi", handleWifi );
server.on ( "/webota", handleWebota );
server.on ( "/help", handleHelp );
server.on ( "/postul2", handlePostUL2 );
server.on ( "/postul2data", handlePostUL2data );
```

**router.ino**
Todas las funciones que renderizan las páginas

```
void handleRoot() {

  String page = FPSTR(HTTP_HEAD);
  page.replace("{v}", "FIWARE_Zone_IoT");
  page += FPSTR(HTTP_STYLE);
  page += FPSTR(HTTP_HEAD_END);
  page += F("<h1>FIWARE IoT Device</h1>");
  page += F("<h3>Menu</h3>");
  page += FPSTR(HTTP_MAIN_FORM);
  page += String("hostname:");
  page += String(dev_hostname);
  page += String(".local");
  page += FPSTR(HTTP_END);
  server.send(200, "text/html", page);

}
```

# 4.4.1 Página de inicio

**router.ino**

const char HTTP_MAIN_FORM[ ]

```
const char HTTP_MAIN_FORM[] PROGMEM  = "<form action=\"/wifi\" method=\"get\"><button>Setup Wifi</button></form><br/><form action=\"/wrst\" method=\"
```

```
const char HTTP_MAIN_FORM[] PROGMEM  =
<form action="/wifi" method="get"><button>Setup Wifi</button></form><br/>
<form action="/wrst" method="get"><button>Clear wifi setup</button></form><br/>
<form action="/webota" method="get"><button>Update</button></form><br/>
<form action="/postul2" method="get"><button>POST UL2</button></form><br/>
<form action="/help" method="get"><button>Help</button></form><br/>
```

**Utilidades de interés:**

Convertidor string C (caracteres de escape)

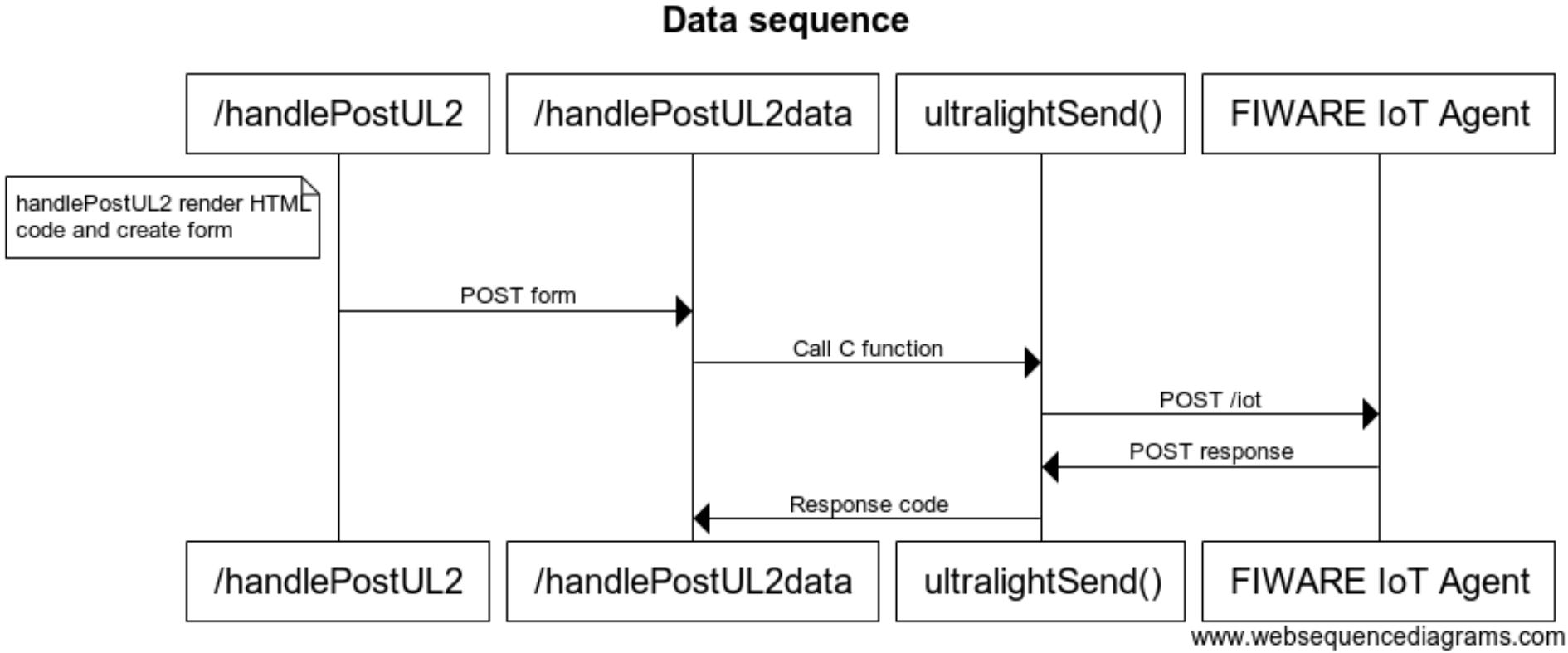
http://tomeko.net/online_tools/cpp_text_escape.php?lang=en

Formateador de HTML

http://minifycode.com/html-beautifier/

# 4.4.1 Página de inicio

# 4.4.2 Envío por UL a FIWARE

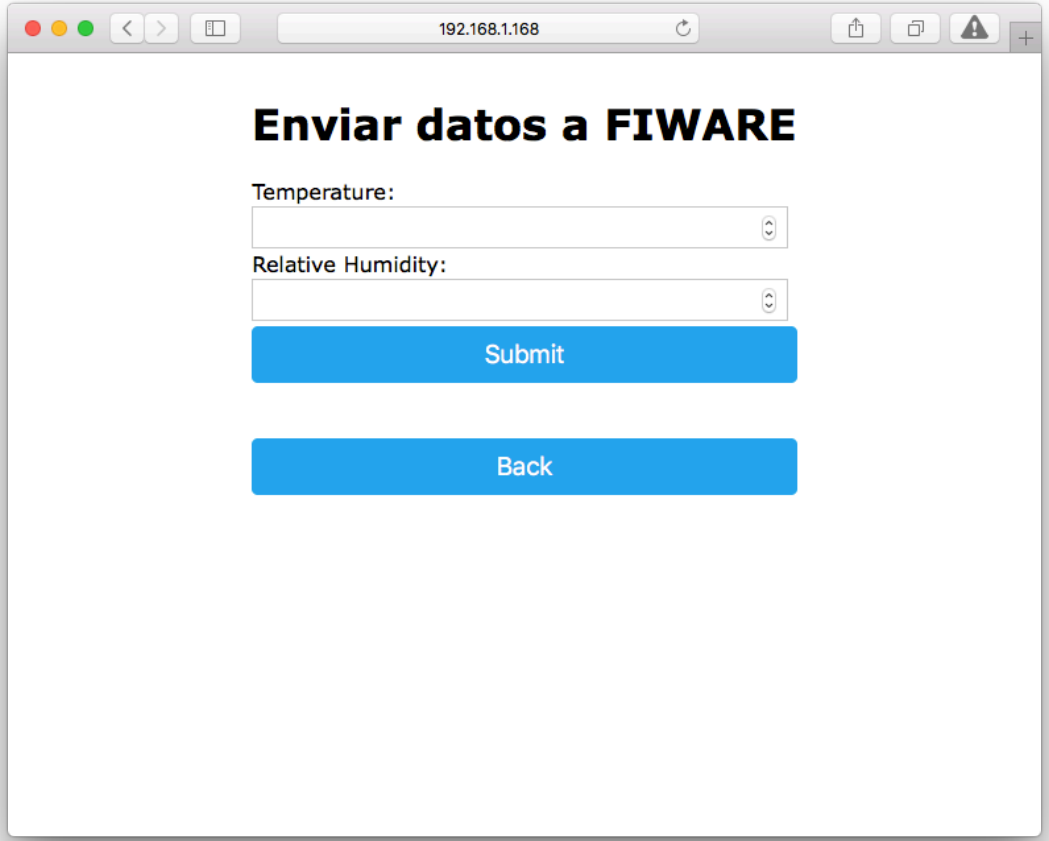


# 4.4.2 Envío por UL a FIWARE

```
void handlePostUL2(){
  String page = FPSTR(HTTP_HEAD);
  page.replace("{v}", "Info");
  page += FPSTR(HTTP_SCRIPT);
  page += FPSTR(HTTP_STYLE);
  page += FPSTR(HTTP_HEAD_END);
  page += F("<h1>Enviar datos a FIWARE</h1>");
  page += F("<form action=\"/postul2data\" method=\"post\">  Temperature: <input type=\"number\
  page += FPSTR(HTTP_BACK_BUTTON);
  page += FPSTR(HTTP_END);
  server.send(200, "text/html", page);
}
```

```
<form action="/postul2data" method="post">Temperature:
    <input type="number" name="tmp" min="-273" max="999">
    <br>Relative Humidity:
    <input type="number" name="rh" min="0" max="100">
    <br>
    <button type="submit" value="Submit">Submit</button>
</form>
```

# 4.4.2 Envío por UL a FIWARE



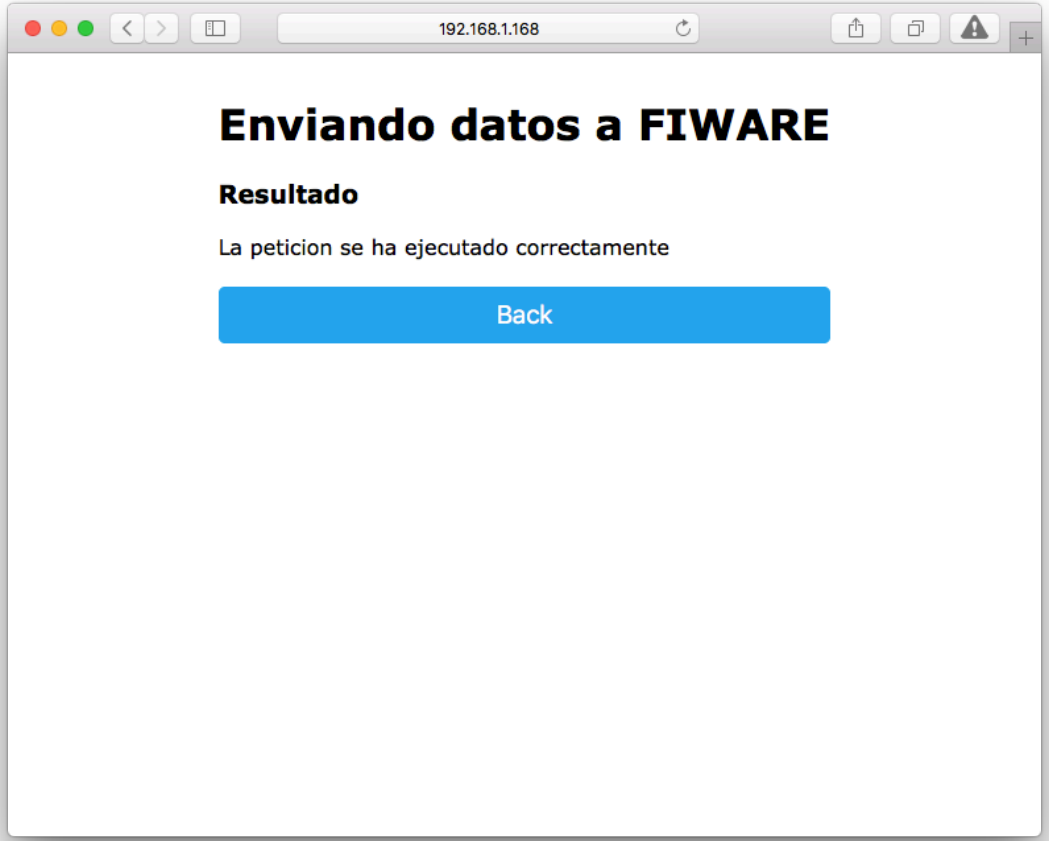
# 4.4.2 Envío por UL a FIWARE

```
void handlePostUL2data(){

    //Search form post values
    String tmp, rh;
      if (server.args() > 0 ) {
          for ( uint8_t i = 0; i < server.args(); i++ ) {
            if (server.argName(i) == "tmp") {
               // do something here with value from server.arg(i);
               tmp = server.arg(i);
            }
            else if(server.argName(i) == "rh"){
              rh = server.arg(i);
            }
          }
      }


    //POST Data
    String page = FPSTR(HTTP_HEAD);
    page.replace("{v}", "Info");
    page += FPSTR(HTTP_SCRIPT);
    page += FPSTR(HTTP_STYLE);
    page += FPSTR(HTTP_HEAD_END);
    page += F("<h1>Enviando datos a FIWARE</h1>");

    //Parse response
    int returnCode = ultralightSend(FIWARE_server,FIWARE_port,FIWARE_token,FIWARE_ID,"temp|"+tmp+"#hr|"+rh);
    if (returnCode ==200){
        page += F("<h3>Resultado</h3>");
        page += F("La peticion se ha ejecutado correctamente");
    }
    else{
        page += F("<h3>Resultado</h3>");
        page += F("Ha habido un error en la peticion");
        page += String(returnCode);
    }

    //End page
    page += FPSTR(HTTP_BACK_BUTTON);
    page += FPSTR(HTTP_END);
    server.send(200, "text/html", page);

}
```

# 4.4.2 Envío por UL a FIWARE

```
int ultralightSend (String URL, String port, String Token, String ID, String Body) {
  int httpCode = 0;

  Serial.println("http://" + URL + ":" + port + "/iot/d?k=" + Token + "&i=" + ID + "&getCmd=1");
  http.begin("http://" + URL + ":" + port + "/iot/d?k=" + Token + "&i=" + ID + "&getCmd=1");          //Specify request destination

  http.addHeader("Content-Type", "text/plain");         //Specify content-type header

  httpCode = http.POST(Body);                           //Send the Body
  String payload = http.getString();                    //Get the response payload

  Serial.println("Request Done");
  Serial.print("Return code: ");
  Serial.println(httpCode);                             //Print HTTP return code
  Serial.print("Response payload: ");
  Serial.println(payload);                              //Print request response payload

  http.end();                                           //Close connection
  return httpCode;
}
```

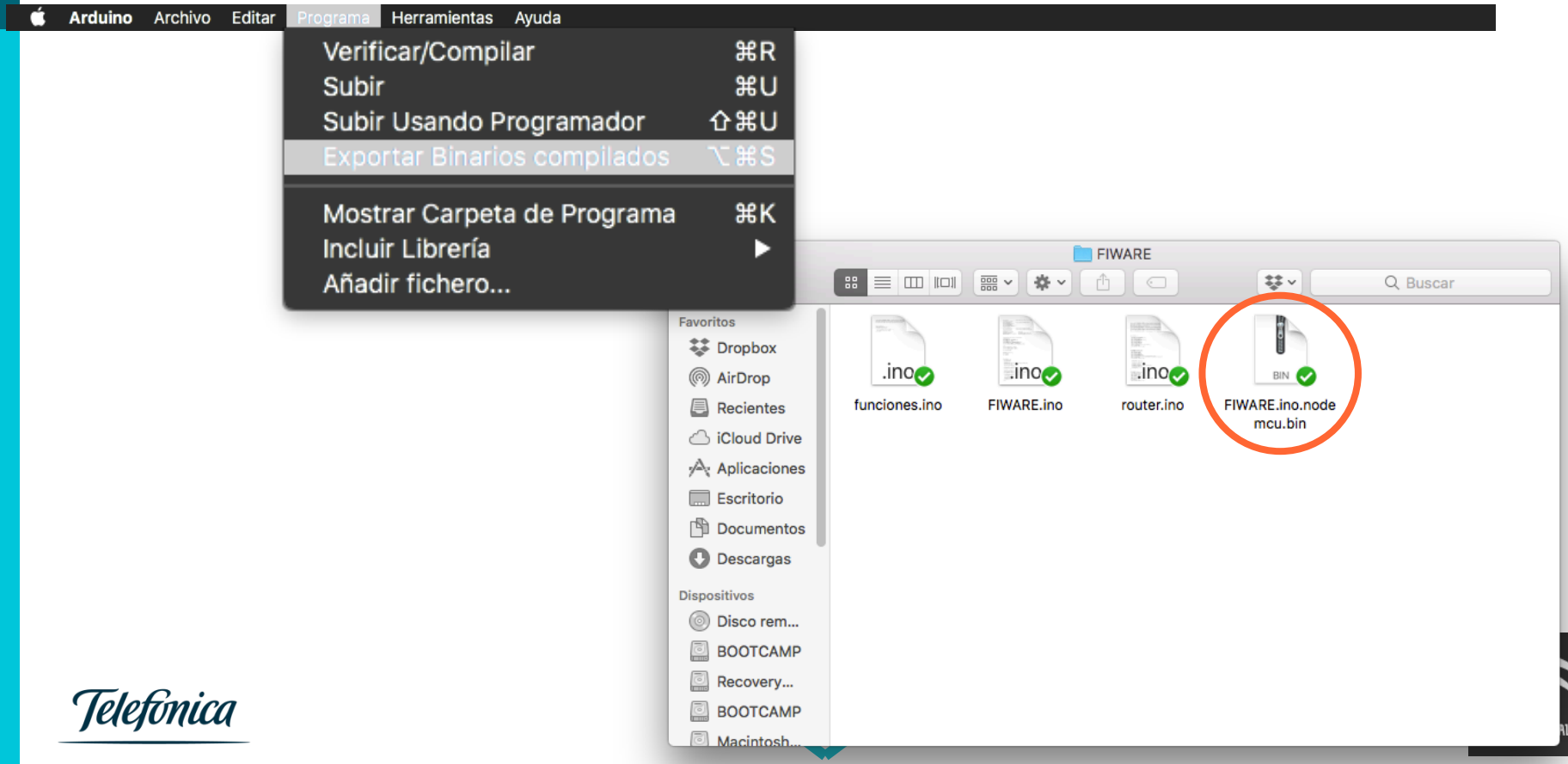
# 4.4.2 Envío por UL a FIWARE



# 4.5 OTA

```
//OTA
const char* update_path = "/webota";
const char* update_username = "admin";
const char* update_password = "admin";
```

# 4.5 OTA


Arduino
Archivo
Editar
Programa
Herramientas
Ayuda
Verificar/Compilar ⌘R
Subir ⌘U
Subir Usando Programador ⇧⌘U
Exportar Binarios compilados ⌥⌘S
Mostrar Carpeta de Programa ⌘K
Incluir Librería
Añadir fichero...
FIWARE
Buscar
Favoritos
Dropbox
AirDrop
Recientes
iCloud Drive
Aplicaciones
Escritorio
Documentos
Descargas
Dispositivos
Disco rem...
BOOTCAMP
Recovery...
BOOTCAMP
funciones.ino
FIWARE.ino
router.ino
FIWARE.ino.node mcu.bin